AMBER

(Accelerator Modeling of Beam with Electrons at Relativistic velocity)

User's Manual

J.-L. Vay, W. Fawley

November 8, 2000

INTRODUCTION

AMBER is a Particle-In-Cell (PIC) code which models the evolution of a representative slice of a relativistic electron beam in a linear accelerator. The beam is modeled as a steady fiow and therefore no electromagnetic waves: all the fields (external and self-fields) are electrostatic and magnetostatic fields (for a complete description, see chapter 5). The possible elements describing the accelerator lattice are solenoids, accelerating gaps, pipes and apertures. Several kinds of beam distribution can be loaded: KV, gaussian, semi-gaussian, etc. Alternatively, the user can reconstruct (or load) a distribution from the output of another codefifor example, an interface generating the beam distribution from output produced from EGUN or LSP codes is available as an option.

This documentation first describes in detail the input files needed to run AMBER and the procedure to start the executable. The possible data files and graphical output are explained in the two following chapters. The last chapter describes the physics model and numerical techniques used. An example of input files and the result obtained with these inputs are also given in the Appendix.

1 RUNNING AMBER

THE COMMAND LINE

To run amber, type the command line

’amber i=<init_filename>l=<lattice_filename>[r=<run_filename> OUTPUT -int -help -search]’

- <init_filename>: filename of the file containing the initialization data,
- <lattice_filename>: filename of the file containing the lattice structure data,
- <run_filename>(optional): a mnemonic to be part of theoutput filename,
- OUTPUT (optional): format for the output file: CGM or POSTSCRIPT. If absent, the default output (screen) is used,
- -int: run in interactive mode (default is batch mode),
- -help <var1> <var2> ...: display definition of <var..> input variable(s),
- -search <word1> <word2> ...: display variable+definition if definition contains <word..>.

Examples:

1. ’amber i=inrun1 l=darhtII.lat r=rundarht1 CGM’

   The input parameters are read from the files ”inrun1” and ”darhtII.lat”. The graphical output will be written in CGM format in the file ’rundarht1.cgm’.

2. ’amber i=inrun2 l=darhtII.lat r=rundarht2’

   The input parameters are read from the files ”inrun2” and ”darhtII.lat”. The graphical output will be displayed on screen.

3. ’amber i=inrun3 l=darhtII_2.lat POST’

   The input parameters are read from the files ”inrun3” and ”darhtII_2.lat”. The graphical output will be written in POSTSCRIPT format in the file ’inrun3.ps’.

THE INITIALIZATION FILES

Three files contain the initial input data for AMBER. All contain FORTRAN namelists which are read by AMBER at the beginning of the run. Two files (’*.lat’ and ’*.mag’) are devoted to the description of the lattice. The other file (’in*’) is general and concerns the description of the beam, as well as test particles or switches for different options.

THE GENERAL PURPOSE INIT FILE (’in*’)

Three namelists are read by AMBER in this file: ’BEAM’, ’BEAM_TEST_NAMELIST’ and

'SIM_PARAM'. The following variables may be set in this namelist:

- 'BEAM':

current	*current in Amps*
energy_mev	*particles (electrons) energy in MeV*
abeam	*outer beam radius in cm*
rprime	*dr/dz at r=abeam*
emit_norm	*normalized emittance*
l_adjust_init_rot	*switch for adjustment of the initial rotation of the beam in accordance with the conservation of angular momentum assuming no Bz on the cathode (default=.true.)*
l_init_pot_depress	*switch for the calculation of the initial space charge potential depression. This lowers the beam kinetic energy on axis from that specified in* **energy_mev** *(default=.true.)*
npart	*number of macroparticles*
xoffset	*beam offset in X (m)*
yoffset	*beam offset in Y (m)*
xpoffset	*beam offset in dX/dZ (rad)*
ypoffset	*beam offset in dY/dZ (rad)*
distribution	*initial distribution, can be 'KV' 'SEMI-GAUSSIAN', 'GAUSSIAN' or 'UNIF_ELLIPSOID' (default=gaussian)*
par_dump_file	*name of the dump file (*.dmp') to dump particles data in at the end of the run (default=NOT SET)*
old_par_dump_file	*name of the dump file to read the initial particles data from. If ='NOT SET" (default), the data are either generated by amber, or read from an EGUN or LSP output.*
dump_type	*type of the dump file saving: can be 'BINARY' or 'ASCII' (default=BINARY)*
l_ReadEgunOutput	*logical switch: reads data from EGUN output if .true. (default=.false.)*
egun_filename	*name of the EGUN output file to read*
l_ReadLSPOutput	*logical switch: reads data from LSP output if .true. (default=.false.)*
LSP_filename	*name of the LSP output file to read*

WARNING: when loading an Egun or a LSP distribution, the distribution will be modified if **l_adjust_init_rot** or **l_init_pot_depress** are **ON**. It is the user responsibility to turn these switches off if the corresponding correction on the distribution is not desired.

density_profile	*density profile multiplier function (of radius r)*
pressure_file	*name of file containing residual pressure data*
f_ion	*constant ion neutralization fraction*
df_ion_dp	*ratio of ionization fraction to pressure in Torr*

- 'BEAM_TEST_NAMELIST':

  Up to nine test particles can be tracked by AMBER and are initialized with the following variables:

npart_test *number of test particles (default=0)*

xtest *X-position in meters {array, size=npart_test}*
ytest *Y-position in meters {array, size=npart_test}*
xvtest *X-speed in m/s {array, size=npart_test}*
yvtest *Y-speed in m/s {array, size=npart_test}*

- 'SIM_PARAM':

Several parameters control the field solver and graphics of a run:

lgraphic *produces graphics output if .true. (default=.true.)*
l_opendx *save OPENDX files if .true. (default=.false.)*

nr *number of grid cells for field calculations (default=500)*

zstart *initial Z location of the beam slice (m) (default=0.)*
zmax *run ends when slice location z>=zmax (default=0.5)*
zstep *step size in Z(m) for particle advance (default=0.05)*

l_paraxial *logical switch: when true, use the paraxial approximation (default=.false.)*
l_ self_er *logical switch: computes self radial electric field if .true. (default=.true.)*
l_ self_ez *logical switch: computes self longitudinal electric field if .true. (enforces l_self_er=.true.) (default=.true.)*
l_ self_br *logical switch: computes self radial magnetic field if .true. (enforces l_self_bz=.true.) (default=.true.)*
l_ self_btheta *logical switch: computes self azimuthal electric field if .true. (default=.true.)*
l_ self_bz *logical switch: computes self longitudinal magnetic (diamagnetic) field if .true. (default=.true.)*
l_self_envelope *logical switch: uses an envelope approxima- for the computation of the beam self-fields 'er' and 'btheta' if true (enforces l_paraxial=.true.) (default=.false.)*
l_image_fields *logical switch: computes image fields if .true.* (*default=.false.*) (this feature is untested!)
l_remap_solgap *logical switch: remaps solenoid and gap fields to accomodate large particcle z-steps if .true. (default=.true.)*

gap_calc_method *method to compute the acceleration gap fieldsfican be 'heaviside' or 'multigrid' (default=multigrid)*
l_gap_er *logical switch: turns ON or OFF the computation of the gap radial electric field (default=.true.)*

AMBER can create four different kinds of phase-space plots.

phaseplot_type	*list of the phase-space plots to display: possible non-exclusive options are 'normal', 'uncor' (no correlations), 'unrot' (no rotations), 'thermal' (no correlations, no rotation), and 'radial'.*
dzphaseplot	*AMBER plots phase-space plots every 'dzphaseplot' meters*
zphaseplot	*AMBER plots phase-space at the zphaseplot locations*
npart_phaseplot	*number of particles to be included in the phase-space plot (these are randomly selected at the beginning of the run)*
phaseplot_save_particles	*logical switch: save particles data for post-processing phase-space plots if .true. (default=.false.)*

For the phase-space snapshot, the ranges of the plot axes can be setup manually by adjusting the following variables (otherwise they are automatically calculated at run time, which is the default). Up to four different sets with different z-ranges can be defined:

phasep_infoN%zrange	*zmin and zmax for Nth phase-space plot serie*
phasep_infoN%xrange	*xmin and zmax for Nth phase-space plot serie*
phasep_infoN%yrange	*ymin and ymax for Nth phase-space plot serie*
phasep_infoN%xprange	*xpmin and xpmax for Nth phase-space plot serie*
phasep_infoN%yprange	*ypmin and ypmax for Nth phase-space plot serie*
phasep_infoN%rrange	*rmin and rmax for Nth phase-space plot serie*
phasep_infoN%rprange	*rpmin and rpmax for Nth phase-space plot serie*
NOTE	N=1..4

l_save_hist	*logical switch: saves history diagnostics in a file if .true. (default=.false.)*
hist_dir	*directory in which to save history files (will be created if does not exist previous to run)*
save_hist_vars	*list of history data variables to save: possible non-exclusive variables are 'r', 'rprime', 'r_four', 'rprime_four', 'energy', 'emittance', 'gamma', 'xyrms', 'xybar', 'eext', 'bext'*
dzhist	*AMBER stores history data every 'dzhist' meters*

zfidplot	*list of locations along z (in meters) where the fields will be plotted*
dzfidplot	*the fields will be plotted every* **dzfidplot** *meters*
nzfidsave	*number of times the fields data will be saved in the file 'fieldsN.dat' during the run (N=10000+snapshot number)*
zfidsave	*list of locations along z (in meters) where the fields will be saved*
nzfidsnapshot	*number of times the fields data will be displayed during the run*
zfidsnapshot	*list of locations along z (in meters) where the fields will be displayed*

THE LATTICE FILE ('*.lat')

This file contains the namelist named 'lat_list' which defines the accelerator lattice with the following

variables:

- name of the input file containing magnet information

  env_magfile: default='NOT SET'

- switch for lengths to be input in centimeters rather than in meters

  l_cm: if .true., then all length in '*.lat' **and** '*.mag' are expressed in cm, in meters otherwise (default=.true.)

- definitions for the different general types of solenoids ({arrays, size_max=6})

soldef_name	*name*
soldef_length	*length*
soldef_r_inner	*inner radius*
soldef_r_outer	*outer radius*
soldef_n_layers	*number of layers*

- definitions for the primary general types of accelerator gaps ({arrays, size_max=6})

gapdef_name	*name*
gapdef_length	*length*
gapdef_rmax	*radius*

- characteristics of the individual gaps in the lattice ({arrays, size_max=128})

gap_name	*name*
gap_type	*type (1..6)*
gap_z	*location in z of the gap centroid*
gap_mev	*gap accelerating 'field' (in MeV)*
gap_offsetX	*individual offset in X (m)*
gap_offsetY	*individual offset in Y (m)*
gap_tiltX	*individual tilt in X (rad)*
gap_tiltY	*individual tilt in Y (rad)*

- characteristics of the beam pipe sections in the lattice

nwall	*number of pipes (max=6)*
rwall	*array(size=nwall): pipes radii (m)*
zwall	*array(size=nwall): pipes starting positions (m)*
wall_offsetX	*array(size=nwall): pipes offsets in X (m)*
wall_offsetY	*array(size=nwall): pipes offsets in Y (m)*

- characteristics of the individual pipe apertures in the lattice

naperture	*number of apertures (max=50)*
shape_aperture	*array(size=nwall): aperture shapes: can be 'square','rectangle','circle' or 'ellipse'*
Xaperture	*array(size=nwall): size in X (m)*
Yaperture	*array(size=nwall): size in Y (m, unused for 'square' or 'circle')*
Zaperture	*array(size=nwall): positions in Z (m)*
aper_offsetX	*array(size=nwall): apertures offsets in X (m)*
aper_offsetY	*array(size=nwall): apertures offsets in Y (m)*

- for the entire lattice

sol_offset_delta	*width of the 2D axisymetric gaussian (at half maximum) for the initialization of random solenoids offsets*
sol_tilt_delta	*width of the 2D axisymetric gaussian (at half maximum) for the initialization of random solenoids tilts*
line_offsetX	*offset in X*
line_offsetY	*offset in Y*
line_tiltX	*tilt in X*
line_tiltY	*tilt in Y*
l**ine_tiltCZ**	*position in Z of the center for the tiltX and tiltY rotations*

THE MAGNETS FILE ('*.mag')

This file contains the namelist named "magnets" describing the individual magnets with the following variables:

- number of magnets in the lattice
  ns
- characteristics of the solenoids in the lattice ({arrays, size_max=128})

solnam	*name*
ltype	solenoid *type (1..6)*
zs	*location in z of the solenoid centroid*
bsi	*number of amp*turns*
izs	*position 'zs(i)' is absolute if izs(i)=0, is relative to preceding magnet position if izs(i)=1*
sol_offsetX	*individual offset in X*
sol_offsetY	*individual offset in Y*
sol_tiltX	*individual tilt in X*
sol_tiltY	*individual tilt in Y*

INTERACTIVE MODE CONTROL

By default, AMBER runs in batch mode, i.e. the user has no control during the run. In interactive mode (command line option **-int**), the control is given to the user before AMBER starts the main loop (after all initializations). This is signaled by the prompt 'AMBER>' appearing on screen. The user can then interact with the running code by entering commands:

- print <var1>, <var2> ...: print values of variables,
- stopat: define breakpoints at which to regain control in interactive mode. Several options are possible:
  - z=<location>: stop at a z location (given in meters),
  - istep=<number of steps>: stop after a number of zteps,
- cont: continue the run (control is given back to the user at the next breakpoint),
- quit: quit the interactive mode and continue the run in batch mode (the control will not be given back to the user, even if breakpoints have been defined),
- exit: exit AMBER.
  This mode is a recent addition to the code and may be buggy - cavent emptor.

2 INITIAL PARTICLE DISTRIBUTION

The initial particle distribution can be either generated at run time or read from an external file. The external file is either a particle dump file created by AMBER at the end of a previous run or a particle dump file created by the particle codes EGUN or LSP. From all the options which can be setup in the initialization namelist, AMBER will choose the first available in the following order:

1. attempt to load an AMBER particle dump filefiif successful, skip 2, 3 and 4,
2. attempt to load an EGUN particle dump filefiif successful, skip 3 and 4,
3. attempt to load a LSP particle dump filefiif successful, skip 4,
4. create particle distribution.

Once one particle distribution has been loaded using one of these methods, it is possible to alter the particle distribution in the x-y space by defining a multiplicative function of the particle distribution density.

> This is done by defining the BEAM namelist varible **density_profile** which is a character string describing a mathematical function of the radius *r*. Any usual mathematical function (sin, cos, exp, ...) can be entered. As in FORTRAN, the decimal logarithm is defined as **log10** while the natural logarithm can be either **log** or **ln**. The power is defined by ****** and the square root writes **sqrt**. An example of valid expression is *density_profile='1.+0.5*(r/0.08)**2'*. Note that the radius *r* is the only independant variable allowedfiany other variable will trigger an error message.

If **l_init_pot_depress** is set to .true., the beam energy is corrected due to the space charge depression as $E = \mathbf{energy\_mev} - 10^{-6} \cdot (\mathbf{current}/4\pi\varepsilon_0 c) * (1 + 2\ln(\frac{R_{wall}}{\mathbf{abeam}}))$.

If **l_adjust_init_rot** is set to .true., AMBER will adjust the individual beam particle canonical angular momenta to cancel the externally applied canonical angular momentum evaluated at z=zstart.

> Because there is a canonical angular momentum (and possible self_B_z) produced by the beam rotation, this process has to be iterative and is obtained by a relaxation procedure. First, the vector potential A_θ produced by the external magnets and by the beam particles rotations, on each particle, is computed. The individual particles rotations velocities are then given by $\vec{u}_\theta = -q\vec{A}_\theta/m_e = e\vec{A}_\theta/m_e$ where $\vec{u}_\theta = \gamma\vec{v}_\theta$, $\gamma = 1/\sqrt{1 - v^2/c^2}$ is the relativistic factor, c is the speed of light, e is the electron charge and m_e is the electron charge. This procedure is iterated 10 times.

WARNING: The user has the responsibility to turn **l_init_pot_depress** and **l_adjust_init_rot** off if needed when reading the particle distribution from an external dump file.

Created by AMBER

The particle distribution is physically defined by the initial beam current, energy, emittance, edge radius, angular deviation, offsets and phase-space distribution. For numerical purpose, it is also necessary to define the number of macroparticles describing the beam. AMBER can initialize four types of distribution (KV, gaussian, semi-gaussian or uniform ellipsoid):

- KV: $n(r) = \frac{1}{\pi a^2}$; $n(p_r) = \frac{1}{\pi p_a^2}$
- gaussian: $n(r) = \frac{1}{\pi a^2}\exp(-\frac{r^2}{a^2})$; $n(p_r) = \frac{1}{\pi p_a^2}\exp(-\frac{p_r^2}{p_a^2})$

- semi-gaussian: $n(r) = \frac{1}{\pi a^2}$; $n(p_r) = \frac{1}{\pi a^2} \exp(-\frac{p_r^2}{p_a^2})$
- uniform ellipsoid (waterbag): $n(r) = \frac{2}{\pi a^2}(1 - \frac{r^2}{a^2})$; $n(p_r) = \frac{2}{\pi p_a^2}(1 - \frac{p_r^2}{p_a^2})$
  where $p = \gamma u = \gamma m_e v$, $a = \mathbf{abeam}$, $p_a = m_e c * \mathbf{emit\_norm}/\mathbf{abeam}$).

From particle dump file

AMBER can read particle dump files created by AMBER, EGUN or LSP. When read from AMBER dump file, the particle distribution is directly imported from the file and setup as the distribution used for the new run. When reading a EGUN or LSP particle dump file, further computation is needed because the description of the distribution is different from what it is in AMBER. In EGUN, the distribution is a collection of 'rays' (typically about 100). This is statistically too poor for a Particle-In-Cell code and AMBER needs to create a distribution with more macroparticles. In LSP, the distribution is described by a set of weighted macroparticles which is not supported in AMBER where all the particles have the same weight. In both cases, the array of the density as a function of radius is generated from the EGUN or LSP distribution. From this density map, a new distribution is generated (bit-reverse in a distorted space to recover the desired density profile) in the real space XY. The other quantities (ux, uy, uz) are then mapped from the EGUN or LSP distribution to the new distribution.

3 DATA FILE OUTPUT

All output data files are saved in the directory **hist_dir**. They are all saved in ASCII format except the dump file which is written in an unformatted binary file.

History files

The possible history output data files are

- *r*: rms radius in mm (=$\sqrt{2}\sqrt{\frac{\sum r^2}{np}}$),
- *rprime*: rms dR/dZ in mrad (=$\sqrt{2}\frac{\sum[(xv_x+yv_y)/v_z]}{np}/\sqrt{\frac{\sum r^2}{np}}$),
- *r_four*: in mm (=3 $\left(\frac{\sum r^4}{np}\right)^{1/4}$),
- *rprime_four*: in mrad (=$\sqrt{2}\frac{\sum\left[r^2(xv_x+yv_y)/v_z\right]}{np}/\left(\frac{\sum r^4}{np}\right)^{3/4}$),
- *xyrms*: x and y rms in mm (=$2\sqrt{\frac{\sum x^2}{np}}$; $2\sqrt{\frac{\sum y^2}{np}}$),
- *xybar*: x and y beam offsets in mm (=$\frac{\sum x}{np}$; $\frac{\sum y}{np}$),
- *energy*: kinetic, potential and total energy in MeV
- *emittance*: emittance in x and y in πmm.mrad. The rms values are given as well as the emittance of the ellipse containing N% of the particles with N=80, 85, 90, 95, 100. The axes and aspect ratios of the ellipses are taken to be that of the rms emittance ellipse.
- *gamma*: average gamma of the particles,
- *eext*: external longitudinal electric field on axis,
- *bext*: external longitudinal magnetic field on axis,

All the files have the suffix **.dat**.

Also, the file **lost_part.dat** is automatically created at each run in the directory **hist_dir**. Each time macroparticles are lost, the number of lost macroparticles and the updated number of remaining macroparticles are recorded in this file, as well as the z-location in the lattice at which the loss occured.

Snapshot files

The snapshot output files are

- partN.dat: contains x(m), y(m), γmv$_x$/mc, γmv$_y$/mc, γmv$_z$/mc and color. The color is an integer which is assigned to each particle at the begining of the run. The particle are sorted in function of their radial position in the beam frame and the color integer is the rank of the particle once reordered. In partN.dat, N is the number of the snapshot+10000. 10000 is added in order to have the file properly ranked in the directory, facilitating the access from another program (e.g. OpenDx).
- fieldsN.dat: contains self and external electrostatic and magnetostatic fields E$_r$, E$_z$, B$_r$, B$_\theta$, B$_z$, E$_{r\ ext}$, R$_{z\ ext}$, B$_{r\ ext}$, B$_{z\ ext}$. N is the number of the snapshot+10000.

Macroparticle dump file

A dump file containing the macroparticle distribution in the x-y-ux-uy-uz space can be saved (in unformatted binary format) at the end of the run. To save it, give **par_dump_file** the dump filename. It will saved in the **hist_dir** directory. To read an old dump file, put the dump filename in **old_par_dump_file** (together with the relative path of the directory in which it is located). There is also an option to write an ASCII dump file in which case the data is arranged in columnar format. To do this, set the input variable ”dump_type” equal to ”ASCII” in the namelist ”beam”.

Opendx description files

In order to read the data files, the IBM open source visualization system OPENDX (see below) needs a description of the data files These descriptions are provided via files written by AMBER with the suffix *.general* and *.dx* which are written in the **hist_dir** directory. For more information about these files, the reader is referred to the OPENDX documentation (see web pages at www.opendx.org).

4 GRAPHICS OUTPUT

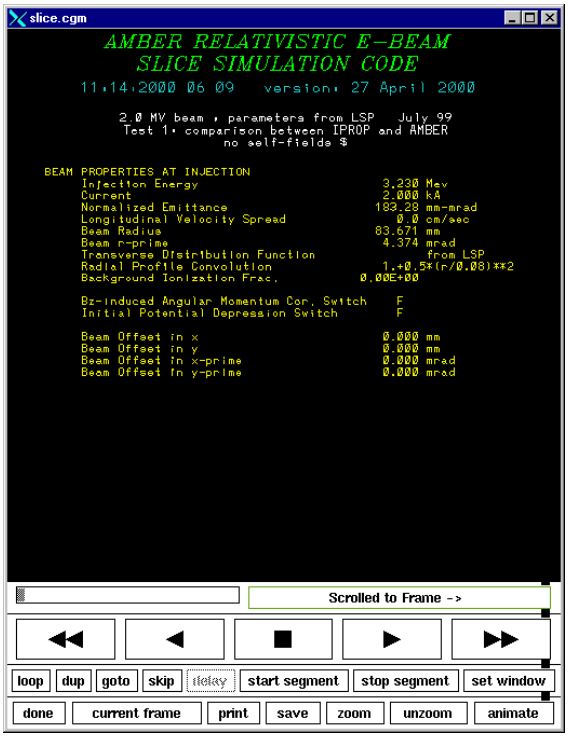
Depending on the platform used to run AMBER, it will use either the NCAR library (on SUN, CRAY, ...) or the DISLIN library (PC with windows, see http://www.linmpi.mpg.de/dislin/) to create output graphics. With the NCAR library, CGM or POSTCRIPT outputs are possible while with the DISLIN libray, only the POSTCRIPT output can be selected. For both, the output can be directed directly to the screen for runtime display without creation of saved graphics file. It is also possible to post-process the output ASCII data files created by AMBER to create graphics. Special files can be created by AMBER for post-processing using the freeware OPENDX (see http://www.opendx.org/) available for several platforms and OS: pc under windows and linux, SUN workstation under SOLARIS (see web pages for others).

NCAR graphics

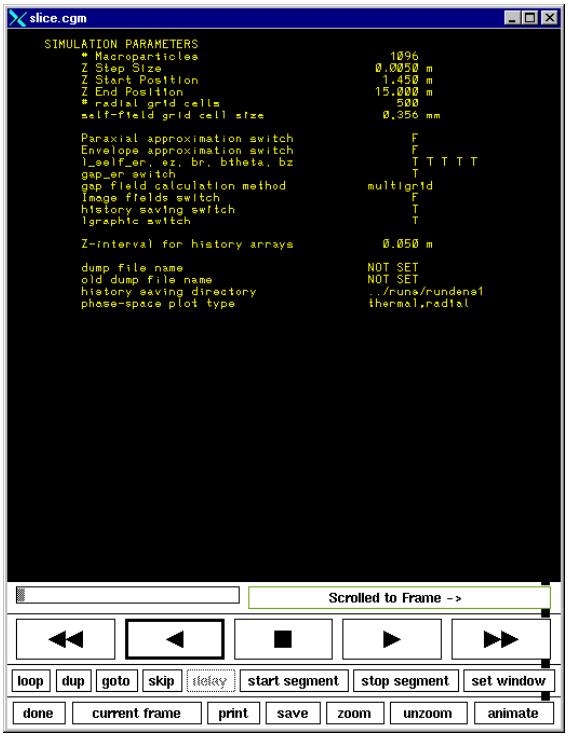
When AMBER is compiled and linked on a UNIX box such as a Sun, NCAR graphics is the only supported form of graphical output (see examples of output on Fig.1 to Fig.6). AMBER relies upon a number of Fortran90 graphics subroutines coded by W. Fawley which are built on top of NCAR and GKS routines. These subroutines are contained in the following F90 source files: fawlib_mod.f90, fawlib90.f90, grf_util90.f90. AMBER currently works with source versions that date from early 2000fi compatibility with either future versions of these routines and/or upgrades to the NCAR graphics library may require additional modifications. Questions may be referred to W. Fawley (WMFawley@lbl.gov).

If the user desires, the NCAR graphics output device/file may be set by optionally adding the uppercase mnemonic *e.g. CGM*. At present, permitted devices include: direct output to an X-windows screen, (mnemonic: X11)fiCGM (computer graphics metafile in NCAR format) file (mnemonic: CGM), color postscript file (mnemonic POST). When running on a UNIX box, the default output is X11 output. If either CGM or POST is selected, the output file will be named ”run_name.cgm” or ”run_name.ps”, respectively, where ”run_name” is set by the -r option on the execute line. For example, *xamber r=run33 POST* will create a postscript file named ”run33.ps”.

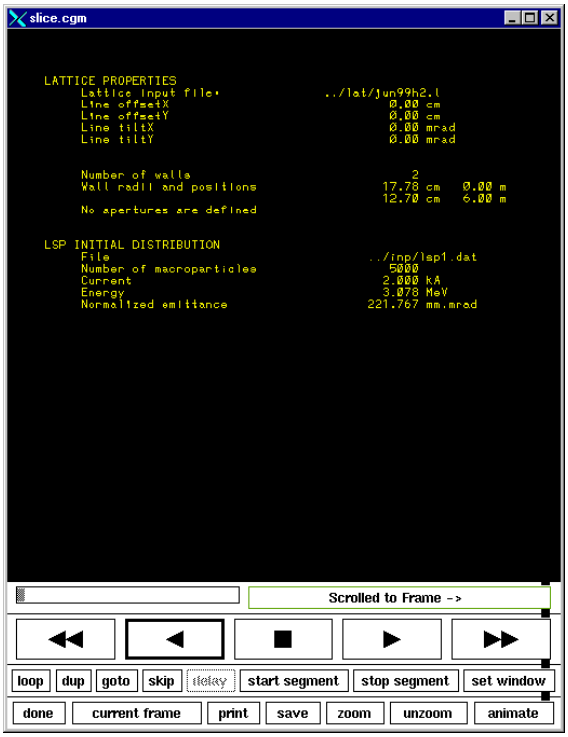
Please note that preexisting files will be overwritten by new runs if the run_name and the output device is the same. The user can use the NCAR family of {**ctrans**} codes to view output CGM files and plot individual frames on various device drivers such as X11, Postscript, *etc.*. The **idt** program is particularly useful on an X-server as one can scroll through a given CGM file, examine multiple frames from one or more CGM files simultaneously, and do some rudimentary animation on screen. The **ghostview** program is useful for examining Postscript output on an X11 screen. Ghostview has the capability to send out individual frames to a printer for hard copy.



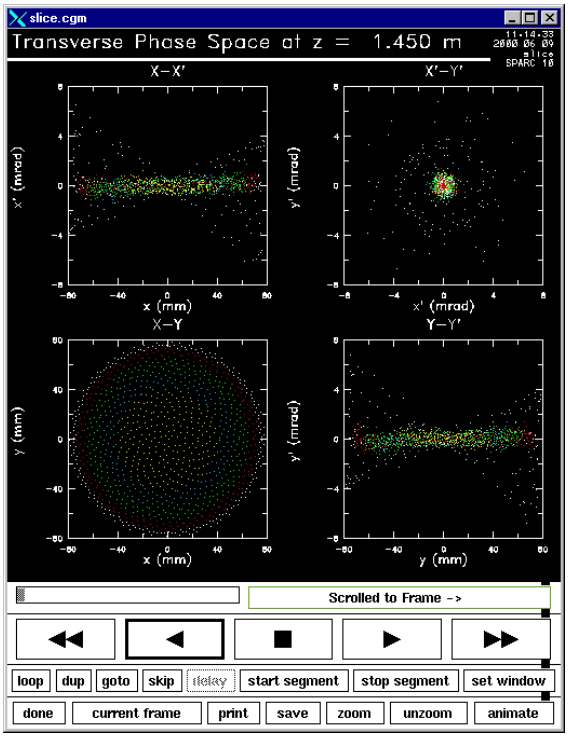
1.IDT wiewer showing NCAR cgm output: front page.



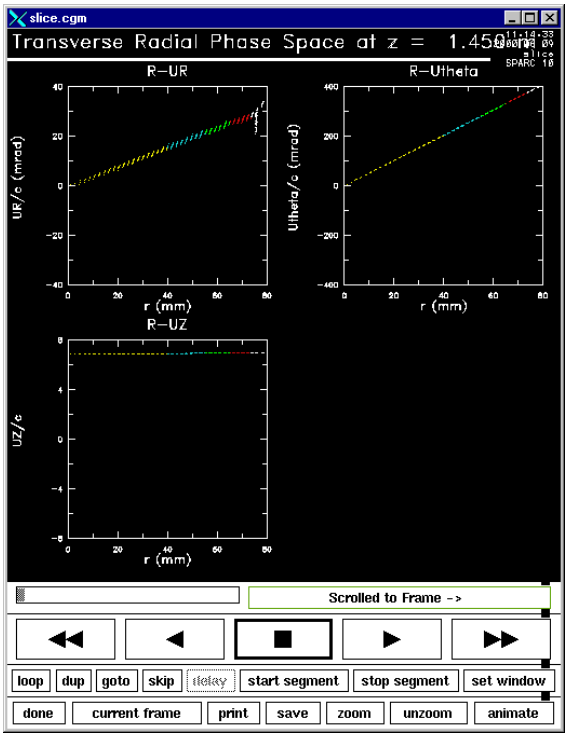
2.IDT wiewer showing NCAR cgm output: second page.



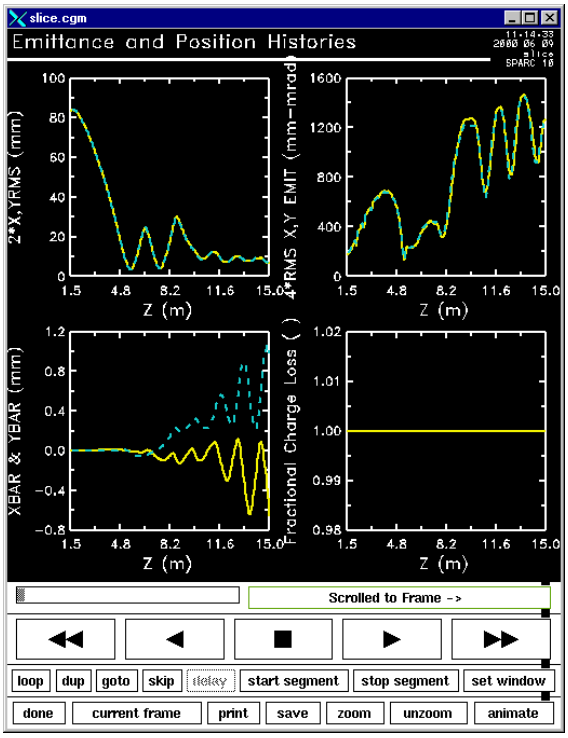
3.IDT wiewer showing NCAR cgm output: third page.



4.IDT wiewer showing NCAR cgm output: phase-space plot (type 'thermal').



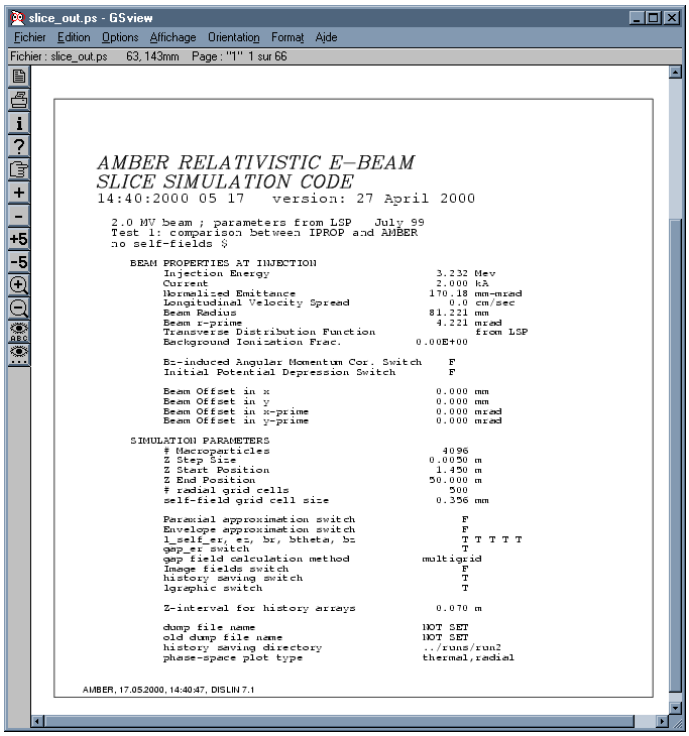
5.IDT wiewer showing NCAR cgm output: phase-space plot (type 'radial').



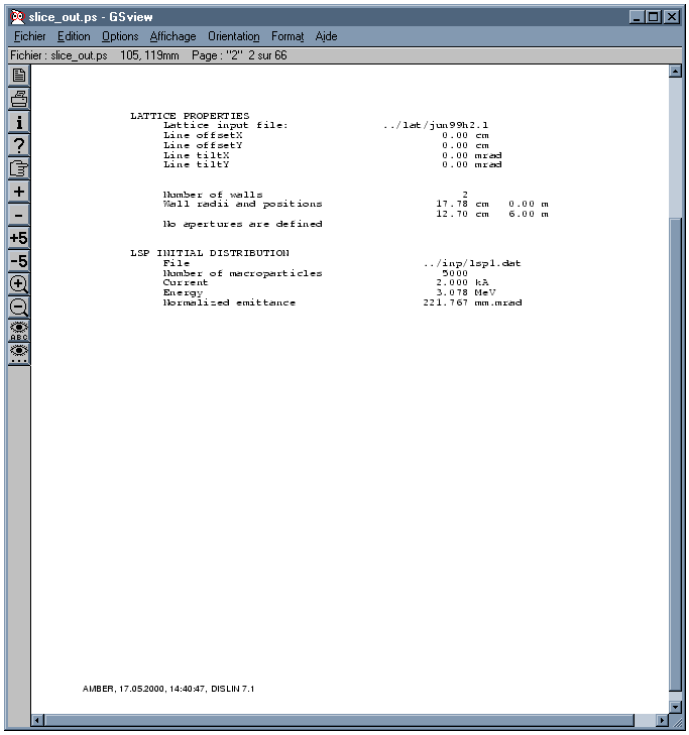
6.IDT wiewer showing NCAR cgm output: emittance and positions histories.

DISLIN graphics (http://www.linmpi.mpg.de/dislin/)

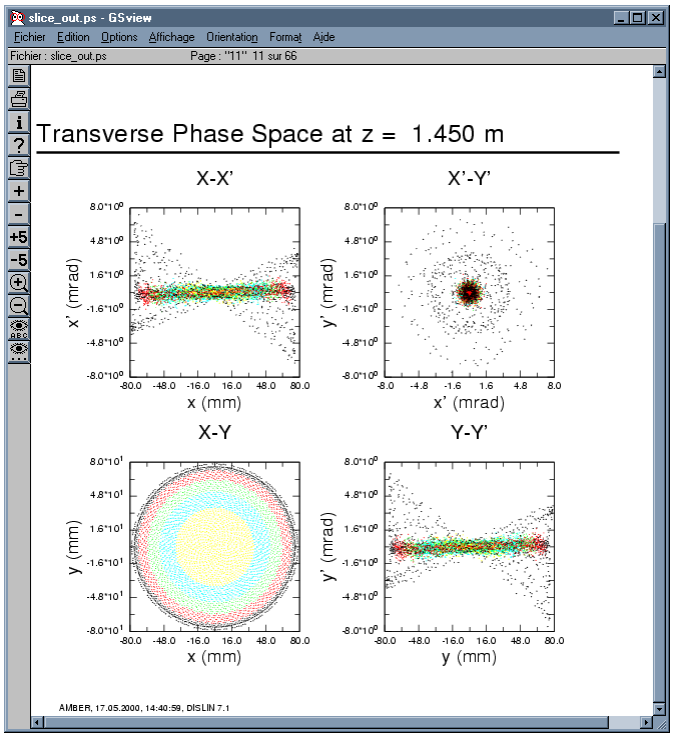
When AMBER is compiled on a PC running under WINDOWS, it uses the DISLIN package for graphical outputs. The possible outputs are the screen (mnemonic WIN) or a color postcript file (mnemonic POST). If a run does not complete, the utility addpg2ps.exe (provided with the AMBER package) has to be run in order to get the paging right in the produced postcript file. By default, the background is white in the postcript file. To reverse the background to black, use the program revps.exe (provided with the AMBER package). Examples of DISLIN output are displayed in Fig.7, 8, 9, 10 and 11.



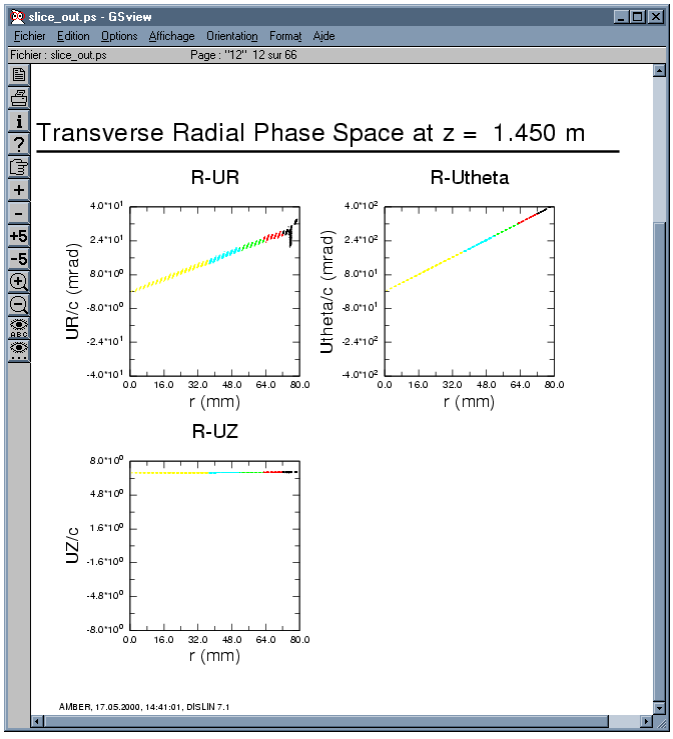
7.Dislin output: front page.



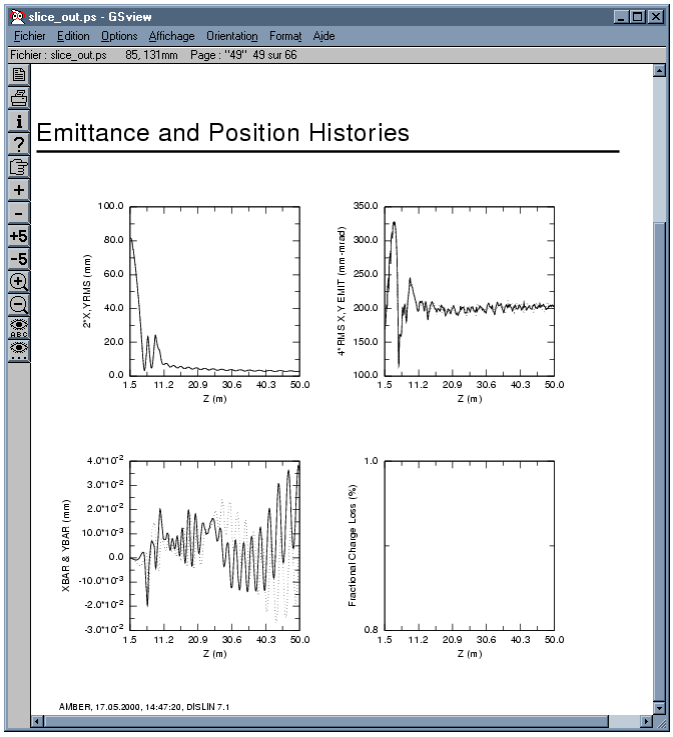
8.Dislin output: second page.



9.Dislin output: phase-space plot (type 'thermal')



10.Dislin output: phase-space plot (type 'radial').



11.Dislin output: emittance and position histories.

OPENDX graphics (http://www.opendx.org)

To use the OPENDX package to analyze AMBER outputs data, the ’opendx’ logical switch must have been set to ’.true.’ in the input files namelist so that required files are written on disk. To view history curves (radius, emittance, ...), use the network ’curves.net’ while to display phase-space plots, use the network ’phasespace.net’. Each of the networks has a graphical user interface. The directory in which to read the data is entered in the box labeled ’Data Directory’. This directory is given relative to the directory from which opendx is executed. When using the networks, the ’Execute’ menu should generally be set to ’Execute on Change’ so that the the program executes each time a change occurs from the user interface (set otherwise if needed). To increase the performance, Opendx caches some results and does not automatically reread from disk data which were already loaded in memory. In case the data have changed on disk and need to be reloaded in memory, a ’Reset Server’ from the ’Connection’ menu must be performed in order to fiush the cache. Then the ’Execute/Execute on Change’ submenu should be selected again.

History curve plotting: network ’curves.net’

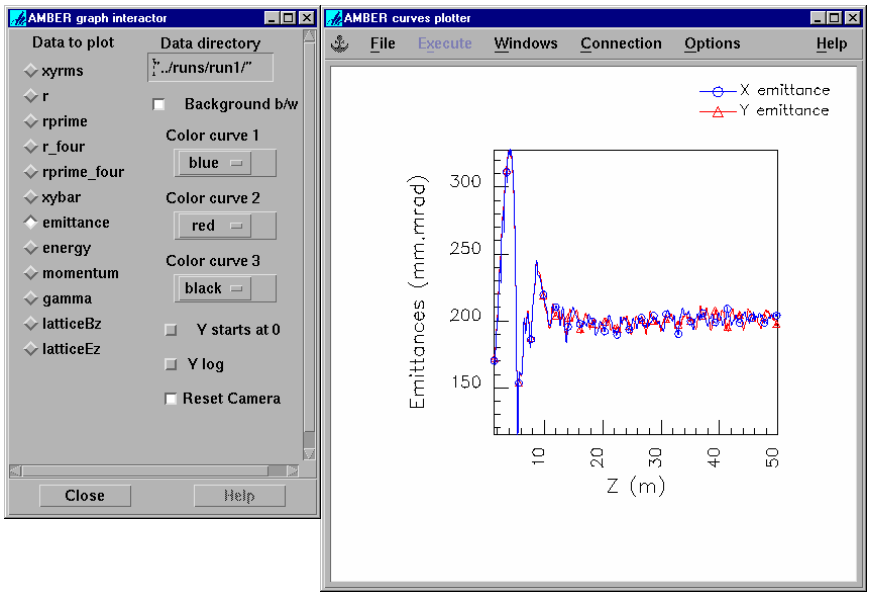
A snapshot of curves.net output is given on Fig.12. On a PC, the curves.net network can be launched via the bat file amberdxc.bat. On another platform, type ’dx -image -program netpath/curves’ where netpath is the path where the curves.net network is. This network displays history curves (xyrms, r, rprime, r_four, rprime_four, xybar, emittance, energy, momentum, gamma) and external field on axis (B_z, E_z). The background color can be set to either black or white with axis and label colors respectively set to white and black. The color of the curves can be set to black, white, yellow, red, green or blue. The Y axis can be forced to start at 0 and one can switch between a linear and a logarithmic y axis (when using the logarithmic scale, the ’Y start at 0’ toggle is disabled). The ’reset camera’ toggle should generally be turned ON unless the user wants to zoom the graph using ’Options/View Control/Zoom’ menu from the ’AMBER curves plotter’ window. The image can be saved in various formats (POSTCRIPT, RGB, tiff, miff, gif, ...) using the ’File/Save Image’ submenu.

Phase-space plotting: network ’phasespace.net’

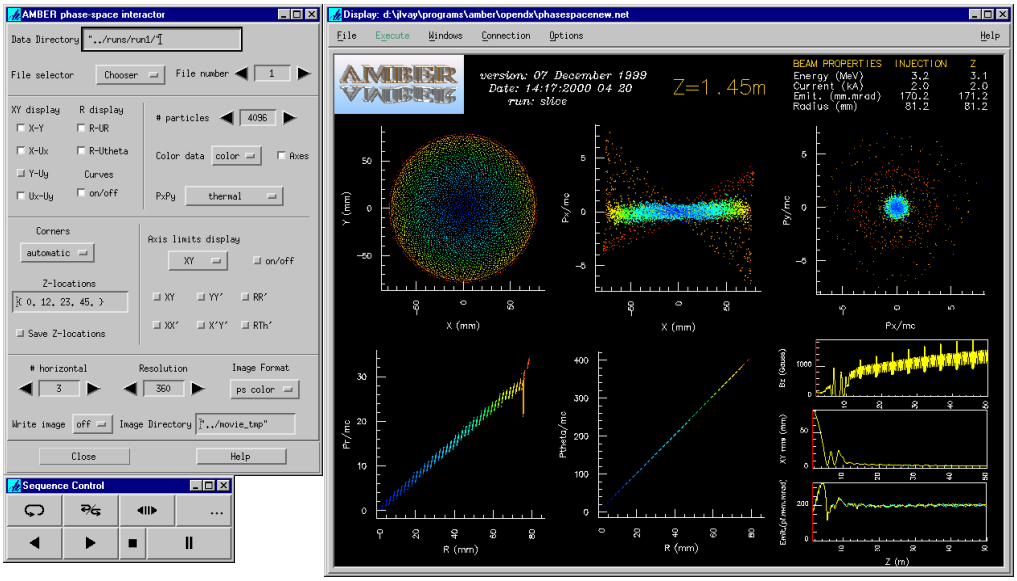
A snapshot of curves.net output is given on Fig.12. On a PC, the curves.net network can be launched via the bat file amberdxp.bat. On another platform, type ’dx -image -program netpath/phasespace’ where netpath is the path where the curves.net network is. If you need more memory than the default accorded on your system for opendx, add the command line option ’-memory N’ where N is the required memory in MB. This network displays various phase-space projections of the particle distribution as well as the external magnetic field on axis, the beam XY rms widths and emittances versus Z. For more explanation, see the help window displayed on Fig.14. This network can be used to make a movie. A quicktime movie was produced by saving miff files using phasespace.net in sequencer mode and using the utility makemovie with the following script:

```
echo ''Convert image files''
FILE=''part?????.miff''
echo ''Images conversion *.miff => *.rgb...''
for i in $FILE
do
echo $i
 imconv $i $i.rgb
done
echo ''Make movie...''
makemovie -f qt -c qt_anim -o darht.mov *.rgb
echo ''Clean .rgb images...''
rm *.*.rgb
echo ''done.''
```

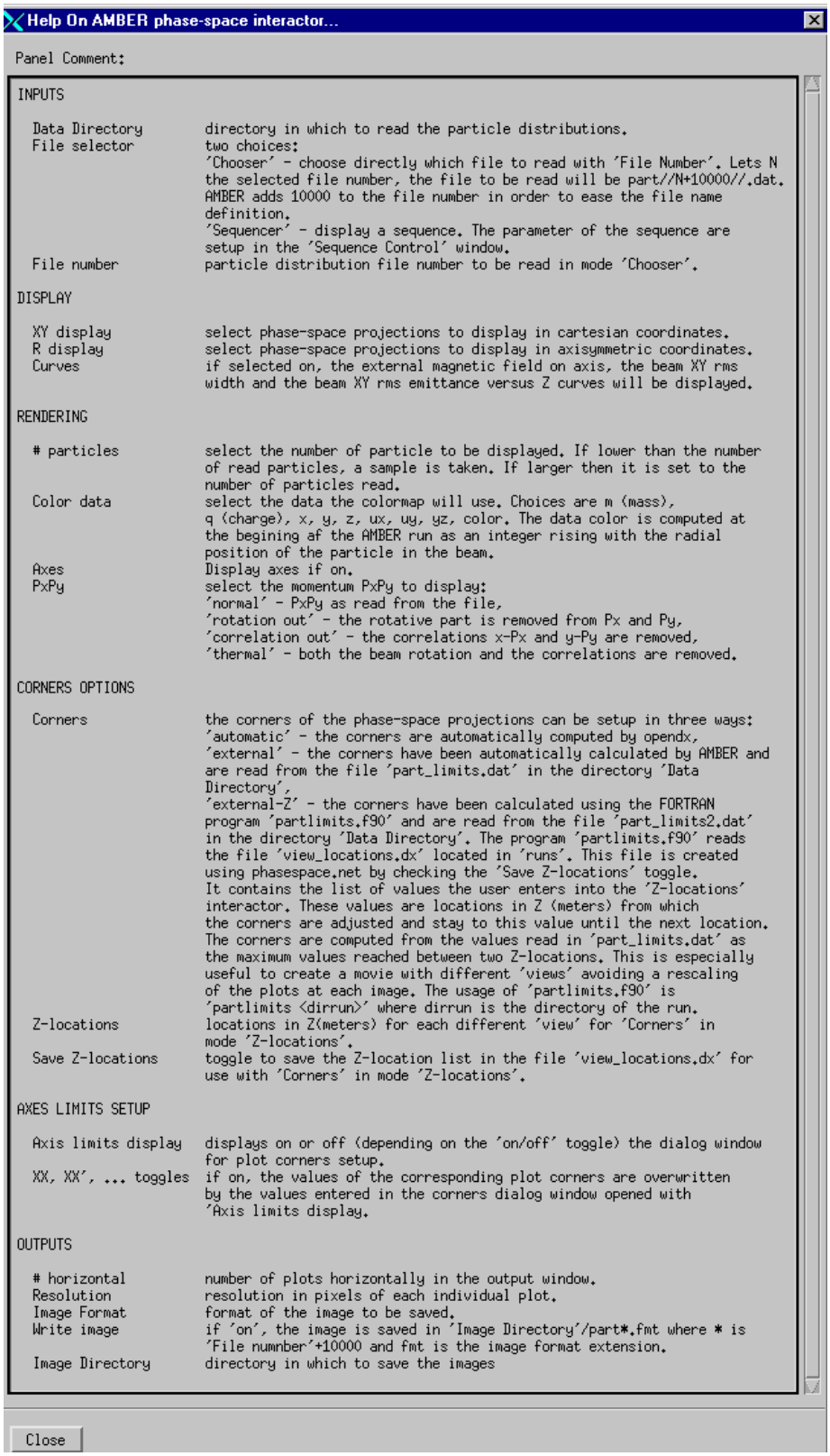
The images are first converted from miff to rgb format and the movie is then created from these rgb files. This conversion is necessary because makemovie does read the miff format (this format was however chosen to save the original images because of its good compression rate).Other utilities can also be used like ’Mediaconvert’ or others.



12.Snapshot of curves.net OPENDX curve plotting.



13.Snapshot of amberdxp.net OPENDX phase-space plotting.



14.OPENDX phasespace.net help file.

5 GENERAL DESCRIPTION

SELF-FIELDS

Physics model

We approximate the beam as an axisymmetric steady fiow having smooth variations in z, so that we can consider:

$$\frac{\partial}{\partial t} = 0$$
$$\frac{\partial}{\partial \theta} = 0$$
$$\frac{\partial}{\partial z} \text{ small}$$

Under these conditions, the Gauss's law gives (assuming $E_z \ll E_r$)

$$\frac{1}{r}\frac{\partial (rE_r)}{\partial r} = \frac{\rho}{\varepsilon_0} \tag{1}$$

and the Ampere's law gives

$$-\frac{\partial B_z}{\partial r} = \mu_0 J_\theta \tag{2}$$
$$\frac{1}{r}\frac{\partial (rB_\theta)}{\partial r} = \mu_0 J_z \tag{3}$$

Knowing the radial electric field and the longitudinal magnetic field, the longitudinal electric field and the radial magnetic field can then be computed using the scalar and vector potentials ϕ and A_θ:

$$\begin{array}{rcl} \phi = -\int E_r dr & \Rightarrow & E_z = -\frac{\partial \phi}{\partial z} \\ A_\theta = -\frac{1}{r}\int r B_z dr & \Rightarrow & B_r = \frac{\partial A_\theta}{\partial z} \end{array} \tag{4}$$

Boundary conditions and image forces: effects of the wall

The boundary conditions at the wall of the pipe are $\phi(r_{wall}) = 0$ and $A_\theta(r_{wall}) = 0$. The second condition gives

$$A_\theta(r_{wall}) = -\frac{1}{r}\int_0^{r_{wall}} r B_z dr = 0 \tag{5}$$
$$= -\frac{1}{r}\int_0^{r_{wall}} r\left(B_z(r_{wall}) + \int_r^{r_{wall}} \mu_0 J_\theta dr\right) dr \tag{6}$$

so that

$$B_z(r_{wall})\int_0^{r_{wall}} r dr = -\int_0^{r_{wall}} r\left(\int_r^{r_{wall}} \mu_0 J_\theta dr\right) dr \tag{7}$$

and

$$B_z\left(r_{wall}\right) = -\frac{\int_0^{r_{wall}} r\left(\int_r^{r_{wall}} \mu_0 J_\theta dr\right) dr}{r_{wall}^2/2} \tag{8}$$

Additionally, image forces are created by the wall. The simplest approximation is an image particle of charge $-Q_b$ (where Q_b is the total charge of the beam), located at $\vec{r} = \frac{r_{wall}^2}{\vec{r}_{offset}}$ (where $\vec{r}_{offset}$ is the offset of the centroid of the beam with respect to the pipe center). This approximation is valid when $r_{offset} \ll r_{wall}$ and $r_{beam} \ll r_{wall}$.

Summary

- electric fields

$$E_r\left(r\right) = \frac{1}{r}\int_0^r \frac{r\rho\left(r\right)}{\varepsilon_0} dr \tag{9}$$

$$\phi\left(r\right) = \int_{r_{wall}}^r E_r\left(r\right) dr \tag{10}$$

$$E_z\left(r\right) = -\frac{\partial\phi\left(r\right)}{\partial z} \tag{11}$$

- magnetic fields

$$B_\theta\left(r\right) = \frac{1}{r}\int_0^r r\mu_0 J_z\left(r\right) dr \tag{12}$$

$$B_z\left(r\right) = \int_{r_{wall}}^r \mu_0 J_\theta\left(r\right) dr - \frac{\int_0^{r_{wall}} r\left(\int_r^{r_{wall}} \mu_0 J_\theta\left(r\right) dr\right) dr}{r_{wall}^2/2} \tag{13}$$

$$A_\theta\left(r\right) = -\frac{1}{r}\int_0^r r B_z\left(r\right) dr \tag{14}$$

$$B_r\left(r\right) = \frac{\partial A_\theta\left(r\right)}{\partial z} \tag{15}$$

- images forces

$$\vec{E}\left(\vec{r}\right) = -\frac{Q_b}{2\pi\varepsilon_0\left(\frac{r_{wall}^2}{\vec{r}_{offset}} - \vec{r}\right)} \tag{16}$$

$$\vec{B}\left(\vec{r}\right) = -\frac{\mu_0 I_b}{2\pi\left(\frac{r_{wall}^2}{\vec{r}_{offset}} - \vec{r}\right)} \times \vec{z} \tag{17}$$

Discretization

The fields are computed on a one dimensional radial grid (index j) every longitudinal step Δz (index i). The discretized equations are then

$$E_r^i\left(j\right) = \frac{1}{r\left(j\right)}\sum_0^{r(j)} \frac{r\left(j-1/2\right)\rho\left(j-1/2\right)}{\varepsilon_0}\Delta r \tag{18}$$

$$\phi^i\left(j\right) = \sum_{r_{wall}}^{r(j)} E_r\left(j+1/2\right)\Delta r \tag{19}$$

$$E_z^i\left(j\right) = -\frac{\left(\phi^i\left(j\right) - \phi^{i-1}\left(j\right)\right)}{\Delta z} \tag{20}$$

and

$$B_\theta^i(j) = \frac{\mu_0}{r(j)} \sum_0^{r(j)} r(j-1/2)\, J_z^i(j-1/2)\, \Delta r \tag{21}$$

$$B_z^i(j) = \mu_0 \sum_{r_{wall}}^{r(j)} J_\theta^i(j+1/2)\, \Delta r \tag{22}$$

$$+ \frac{\mu_0 \Delta r^2}{r_{wall}^2/2} \sum_0^{r_{wall}} r(j-1/2) \left(\sum_{r(j)}^{r_{wall}} J_\theta^i(j-1/2) \right) \tag{23}$$

$$A_\theta^i(j) = -\frac{1}{r(j)} \sum_0^{r(j)} r(j-1/2)\, B_z^i(j-1/2)\, \Delta r \tag{24}$$

$$B_r^i(j) = \frac{\left(A_\theta^i(j) - A_\theta^{i-1}(j)\right)}{\Delta z} \tag{25}$$

Note that the calculation of B_z^i is split into three steps: first the calculation of B_z without the wall, then the calculation of B_z at the wall and finally the correction of B_z due to the wall.

A comparison of the self-fields calculated by AMBER are compared to analytical results for a KV beam in Appendix A.

EXTERNAL FIELDS

Solenoidal

The solenoids are modeled either as N_l infinitely thin coaxial layers of wires ('thin approximation') positioned regularly between r_{in} (inner radius) and r_{out} (outer radius), or as one thick layer of thickness $\delta r = r_{out} - r_{in}$ ('thick approximation').The magnetic field is computed on a cartesian uniform bidimensional RZ grid. It is obtained by a series expansion of the analytic solution on axis computed for each layer with the selected approximation (thin or thick). The solution on axis for one layer of length L and radius r is given as a function of z by

- thin approximation

$$B_z(0,z) = \frac{\mu_0 I}{2 N_l L} \left(\frac{z_p^2}{\sqrt{r^2 + z_p^2}} - \frac{z_m^2}{\sqrt{r^2 + z_m^2}} \right) \tag{26}$$

- thick approximation

$$B_z(0,z) = \frac{\mu_0 I}{2\delta r L} \left(\begin{array}{c} \left[z_p \ln\left(r_{out} + \sqrt{r_{out}^2 + z_p^2} \right) - z_m \ln\left(r_{out} + \sqrt{r_{out}^2 + z_m^2} \right) \right] \\ - \left[z_p \ln\left(r_{in} + \sqrt{r_{in}^2 + z_p^2} \right) - z_m \ln\left(r_{in} + \sqrt{r_{in}^2 + z_m^2} \right) \right] \end{array} \right) \tag{27}$$

where I is the number of amp.turns and

$$z_p = z + 0.5L \tag{28}$$

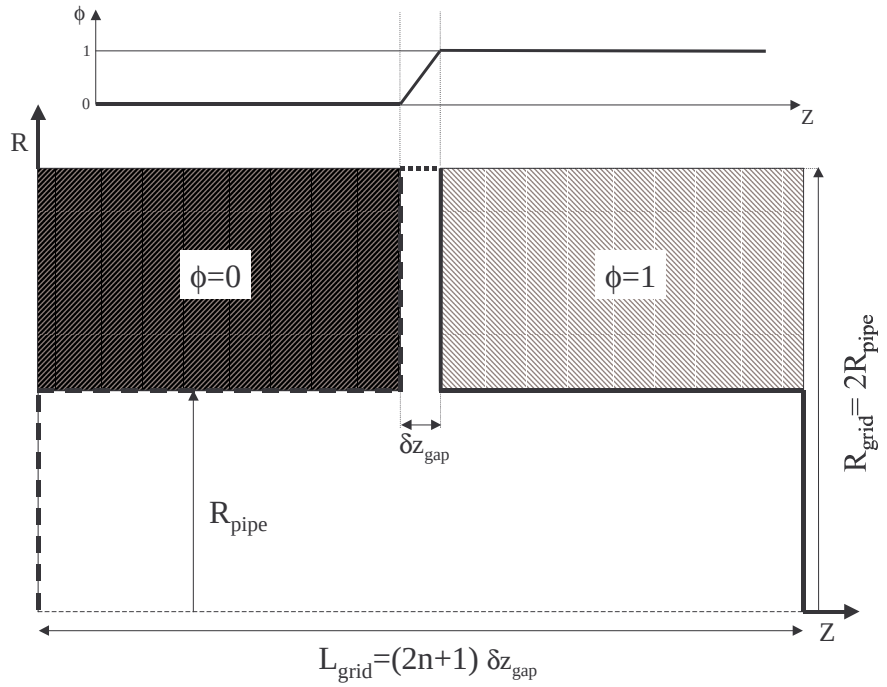
$$z_m = z - 0.5L \tag{29}$$

The radial and longitudinal fields are then given by the serie expansion

$$B_r(r,z) = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!\,(n-1)!} \frac{\partial^{2n-1} B_z(0,z)}{\partial z^{2n-1}} \left(\frac{r}{2}\right)^{2n-1} \tag{30}$$

$$B_z(r,z) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(n!)^2} \frac{\partial^{2n} B_z(0,z)}{\partial z^{2n}} \left(\frac{r}{2}\right)^{2n} \tag{31}$$

The derivatives $\frac{\partial^i B_z(0,z)}{\partial z^i}$ were derived analytically using Mathematica for $1 \leq i \leq 10$.



15.Geometry of the problem. The system is axisymmetric in RZ coordinates. The box of computation is a R_{grid} x L_{grid} rectangle with R_{grid}=2R_{pipe} and L_{grid}=(2n+1)δz_{gap}. The boundary conditions are: $\phi = 0$ along the thick dashed linefi$\phi = 1$ along the thick plain linefiϕ is linearly interpolated along the thick dotted line.

Gap electric

The calculation of the gap fields utilizes a multigrid solver (for more details, see [1], [2] and [3]). An example of the application of the gap field solver in AMBER is given in Appendix B.

Geometry

The system is axisymmetric 'RZ' and has a shape as described by figure 15. Defining R_{pipe} to be the radius of the pipe and δz_{gap} to be the length of the gap, we define a box of computation of radius R_{grid}=2R_{gap} and of length $L_{grid} = (2n+1)\,\delta z_{gap}$ (where n is an integer being chosen so that L_{grid} is as close as possible of the desired spatial extension of the gap field computation in the longitudinal direction).

In order to maximize both efficiency and accuracy, the computation was divided in two steps. A first step computes the solution for the part of the grid extending from r=0 to r=R_{pipe} only. The simulation region is then a rectangular box with straightforward boundary condition where we can begin the (full) multigrid calculation on a very coarse grid. As the solution grid becomes finer and finer, we reach the second step of the calculation when the grid happens to be fine enough to resolve δz_{gap}. The grid is then redefined to fit to the geometry of the problem so that the gap boundaries map onto lines of the grid and corners onto nodes. This explain our choice of 2R_{gap} x $(2n+1)\,\delta z_{gap}$ as the dimensions of the grid. While the first step enhances the efficiency of the multigrid algorithm by allowing the computation on very coarse grid, the second step enhances the accuracy by placing the boundaries of the gap on the grid, avoiding interpolations between grid points for the boundary conditions.

Boundary conditions

The difference of potential between the two gap conductors is normalized to one and the corresponding potential values are assigned as boundary values on the boundaries of the conductor. Between the two conductors, on the large radius side of the grid, the boundary value is assumed to follow a linear progression

between the two conductors. This boundary condition is translated from r=2R_{pipe} to r=R_{pipe} when performing the first step of the calculation, as described in the preceding section. At each end of the grid, the radial electric field E_r is assumed to vanish. This gives a constant potential (which is the one of the corresponding conductor at each end) as the boundary condition at these locations. This assumption of E_r vanishing at both ends corresponds to an infinite system (in the longitudinal direction) of gaps regularly spaced by $(2n+1)\,\delta z_{gap}$. When L_{grid} becomes sufficiently large, both E_r and E_z vanish at the boundaries which corresponds to an isolated gap.

Large step remapping

The solenoidal and gap fields are known on a discrete 2D-RZ cartesian grid and the external field is deposited onto each particle by linear interpolation. If the particle mover Δz step is smaller than the external field grid, then the particle experiences the totality of the external field as the beam progresses through the lattice. If it is larger, then there will be a sampling in z (possibly but less likely in r) of the external field. Concerning the gap electric fields for example, the particles would then get less acceleration than required. In order to circumvant this problem, the external fields are remapped on grids having Δz larger than the Δz of the particle mover, if necessary, during the initialization process of the run.

NOTE: to activate this option, set **l_remap_solgap**=.true. in the SIM_PARAM namelist.

THE PARTICLE MOVER

The mover is a relativistic Leap-Frog Boris mover as described in [4], p.356. The particles are advanced from one location z to a new one z+Δz, as following (i:time index):

o	+	o	+	o	+	o	+	o
$\overrightarrow{v}^{i-2}$	$\overrightarrow{x}$	$\overrightarrow{v}^{i-1}$	$\overrightarrow{x}$	$\overrightarrow{v}^{i}$	$\overrightarrow{x}$	$\overrightarrow{v}^{i+1}$	$\overrightarrow{x}$	$\overrightarrow{v}^{i+2}$
.	z	.	$z+\Delta z$	.	$z+2\Delta z$	.	$z+3\Delta z$	.
Δt^{i-2}	|	Δt^{i-1}	|	Δt^{i}	|	Δt^{i+1}	|	Δt^{i+2}

All the particles are advanced from the same z location to the same new one. However, they have different velocities. Hence, each particle is advanced in time according to its own time step $\Delta t^i = \Delta z / v_z^i$ (updated at each iteration). In other words, the n[th] particles is advanced according to

- $x^i(n) = x^{i-1}(n) + \Delta t^{i-1/2}(n) v_x^{i-1/2}(n)$
  $y^i(n) = y^{i-1}(n) + \Delta t^{i-1/2}(n) v_y^{i-1/2}(n)$
  $z^i = z^{i-1} + \Delta z$
- $\vec{v}^{i+1/2}(n) = \vec{v}^{i-1/2}(n) + \Delta t^{i-1/2}(n) \vec{F}^i(n)/m(n)$ (using Boris scheme)
- $\Delta t^{i+1/2}(n) = \Delta z / \vec{v}^{i+1/2}(n)$
  where $\vec{F}^i(n)$ is the force acting on the particle.
  The resulting algorithm is fully explicit, at the expense of being slightly off-centered.

A Comparison of the AMBER self-field solver with the analytic solution for a KV beam

We have

- electric fields

$$E_r(r) = \frac{1}{r}\int_0^r \frac{r\rho(r)}{\varepsilon_0} dr \tag{32}$$

$$\phi(r) = \int_{r_{wall}}^r E_r(r)\, dr \tag{33}$$

$$E_z(r) = -\frac{\partial \phi(r)}{\partial z} \tag{34}$$

- magnetic fields

$$B_\theta(r) = \frac{1}{r}\int_0^r r\mu_0 J_z(r)\, dr \tag{35}$$

$$B_z(r) = \int_{r_{wall}}^r \mu_0 J_\theta(r)\, dr - \frac{\int_0^{r_{wall}} r\left(\int_r^{r_{wall}} \mu_0 J_\theta(r)\, dr\right) dr}{r_{wall}^2/2} \tag{36}$$

$$A_\theta(r) = -\frac{1}{r}\int_0^r r B_z(r)\, dr \tag{37}$$

$$B_r(r) = \frac{\partial A_\theta(r)}{\partial z} \tag{38}$$

and

$$\omega_c = \frac{eB_0}{2\gamma m_e} \tag{39}$$

$$\rho(r) = \frac{I}{\pi r_b^2 v_z} \tag{40}$$

$$J_\theta(r) = \frac{I}{\pi r_b^2 v_z} v_\theta = \frac{I}{\pi r_b^2 v_z}\omega_c r \tag{41}$$

$$J_z(r) = \frac{I}{\pi r_b^2} \tag{42}$$

so that

$$E_r(r) = \frac{1}{r}\int_0^r \frac{r\rho(r)}{\varepsilon_0} dr \tag{43}$$

$$= \frac{Ir}{2\pi\varepsilon_0 r_b^2 v_z} \quad r \le r_b \tag{44}$$

$$= \frac{I}{2\pi\varepsilon_0 r v_z} \quad r \ge r_b \tag{45}$$

and

$$B_\theta(r) = \frac{1}{r}\int_0^r r\mu_0 J_z(r)\,dr \tag{46}$$
$$= \frac{\mu_0 I r}{2\pi r_b^2} \quad r \le r_b \tag{47}$$
$$= \frac{\mu_0 I}{2\pi r} \quad r \ge r_b \tag{48}$$

This gives

$$\phi(r) = \int_r^{r_{wall}} E_r(r)\,dr \tag{49}$$
$$= \int_{r_b}^{r_{wall}} E_r(r)\,dr + \int_r^{r_b} E_r(r)\,dr = \frac{I}{2\pi\varepsilon_0 v_z}\left(\ln\frac{r_w}{r_b} + \frac{r_b^2 - r^2}{2r_b^2}\right) \quad r \le r_b \tag{50}$$
$$= \frac{I}{2\pi\varepsilon_0 v_z}\ln\frac{r_w}{r} \quad r \ge r_b \tag{51}$$

and

$$E_z(r) = -\frac{\partial\phi(r)}{\partial z} \tag{52}$$
$$= -\frac{\partial\phi(r)}{\partial r_b}\frac{\partial r_b}{\partial z} = -\frac{\partial\phi(r)}{\partial r_b}\frac{v_{rb}}{v_z} \tag{53}$$
$$= \frac{I v_{rb}}{2\pi\varepsilon_0 r_b v_z^2}\left(1 - \frac{r^2}{r_b^2}\right) \quad r \le r_b \tag{54}$$
$$= 0 \quad r \ge r_b \tag{55}$$

We also have

$$B_z(r) = \int_{r_{wall}}^r \mu_0 J_\theta(r)\,dr - \frac{\int_0^{r_w} r\left(\int_r^{r_w}\mu_0 J_\theta(r)\,dr\right)dr}{r_w^2/2} \tag{56}$$
$$= \int_{r_b}^r \mu_0 J_\theta(r)\,dr + \frac{\int_0^{r_b} r\left(\int_{r_b}^r \mu_0 J_\theta(r)\,dr\right)dr}{r_w^2/2} \tag{57}$$
$$= \frac{\mu_0 I\omega_c}{2\pi v_z}\left(\frac{r^2}{r_b^2} - 1 + \frac{r_b^2}{2r_w^2}\right) \quad r \le r_b \tag{58}$$
$$= \frac{\mu_0 I\omega_c r_b^2}{4\pi v_z r_w^2} \quad r \ge r_b \tag{59}$$

giving

$$A_\theta(r) = -\frac{1}{r}\int_0^r r B_z(r)\,dr \tag{60}$$
$$= \frac{\mu_0 I\omega_c r}{4\pi v_z}\left(1 - \frac{r_b^2}{2r_w^2} - \frac{r^2}{2r_b^2}\right) \quad r \le r_b \tag{61}$$
$$= \frac{\mu_0 I\omega_c r_b^2}{8\pi v_z}\left(\frac{1}{r} - \frac{r}{r_w^2}\right) \quad r \ge r_b \tag{62}$$

and

$$B_r(r) = \frac{\partial A_\theta(r)}{\partial z} \tag{63}$$

$$= \frac{\partial A_\theta(r)}{\partial r_b}\frac{\partial r_b}{\partial z} = \frac{\partial A_\theta(r)}{\partial r_b}\frac{v_{rb}}{v_z} \tag{64}$$

$$= \frac{\mu_0 I \omega_c v_{rb}}{4\pi v_z^2}\left(\frac{r^3}{r_b^3} - \frac{r_b r}{r_w^2}\right) \quad r \le r_b \tag{65}$$

$$= \frac{\mu_0 I \omega_c v_{rb} r_b}{4\pi v_z^2}\left(\frac{1}{r} - \frac{r}{r_w^2}\right) \quad r \ge r_b \tag{66}$$

Summary

- electric fields

$$E_r(r) = \frac{Ir}{2\pi\varepsilon_0 r_b^2 v_z} \quad r \le r_b \tag{67}$$

$$= \frac{I}{2\pi\varepsilon_0 r v_z} \quad r \ge r_b \tag{68}$$

$$E_z(r) = \frac{I v_{rb}}{2\pi\varepsilon_0 r_b v_z^2}\left(\frac{r^2}{r_b^2} - 1\right) \quad r \le r_b \tag{69}$$

$$= 0 \quad r \ge r_b \tag{70}$$

- magnetic fields

$$B_r(r) = \frac{\mu_0 I \omega_c v_{rb}}{4\pi v_z^2}\left(\frac{r^3}{r_b^3} - \frac{r_b r}{r_w^2}\right) \quad r \le r_b \tag{71}$$

$$= \frac{\mu_0 I \omega_c v_{rb} r_b}{4\pi v_z^2}\left(\frac{1}{r} - \frac{r}{r_w^2}\right) \quad r \ge r_b \tag{72}$$

$$B_\theta(r) = \frac{\mu_0 I r}{2\pi r_b^2} \quad r \le r_b \tag{73}$$

$$= \frac{\mu_0 I}{2\pi r} \quad r \ge r_b \tag{74}$$

$$B_z(r) = \frac{\mu_0 I \omega_c}{2\pi v_z}\left(\frac{r^2}{r_b^2} - 1 + \frac{r_b^2}{2r_w^2}\right) \quad r \le r_b \tag{75}$$

$$= \frac{\mu_0 I \omega_c r_b^2}{4\pi v_z r_w^2} \quad r \ge r_b \tag{76}$$

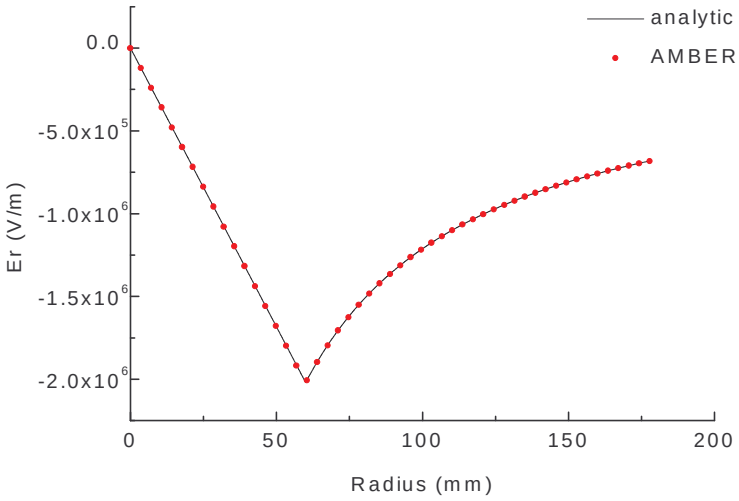
Comparison with AMBER

The initial parameters are

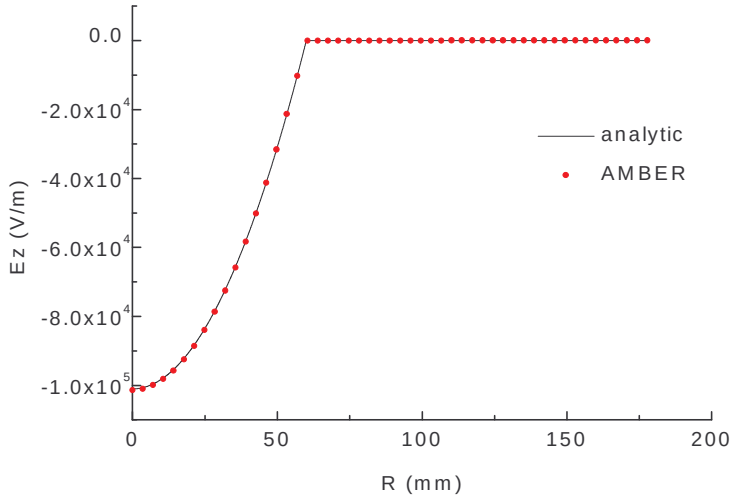
Energy	=	3 MeV
I	=	2 kA
r_b	=	6 cm
r_w	=	17.78 cm
$\frac{v_{rb}}{v_z}$	=	-50 mrad
B_0	=	200 Gauss

The comparisons for E_r, E_z, B_r, B_θ and B_z are respectively displayed on Fig.16, 17, 18, 19 and 20. Some of the corresponding normalized forces acting on one particle are then displayed on Fig.21.

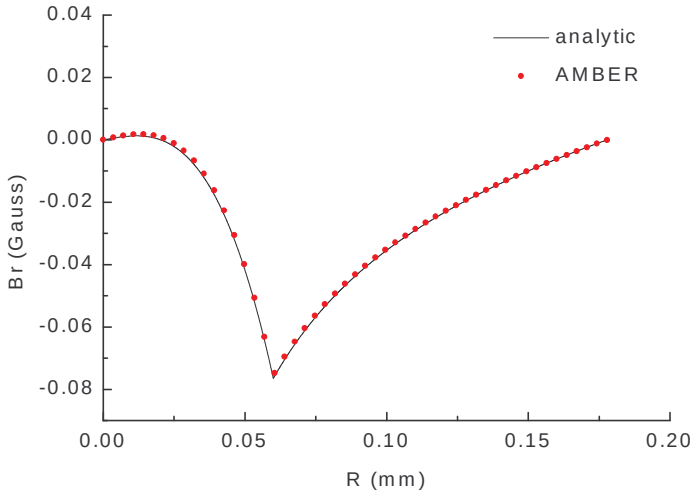
NOTE: despite its low value, the self B_r has a non-negligible effect on the particles.



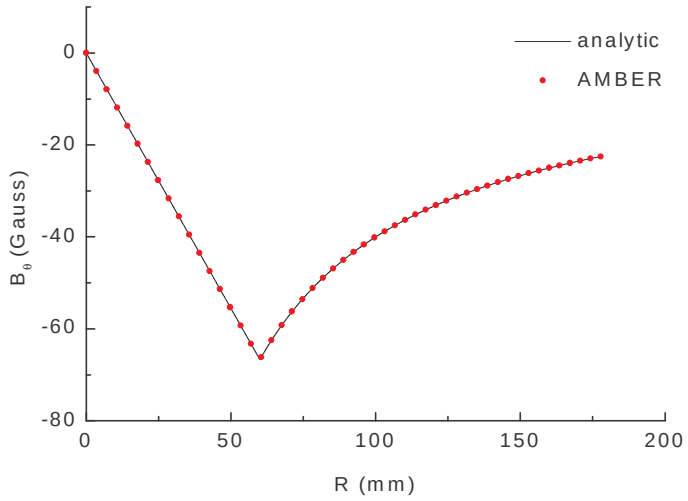
16.Radial electric self-field for a KV beam.



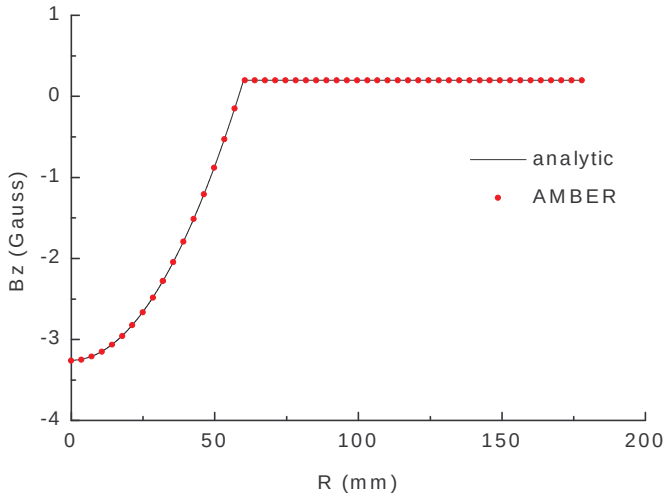
17.Longitudinal electric self-field for a KV beam.



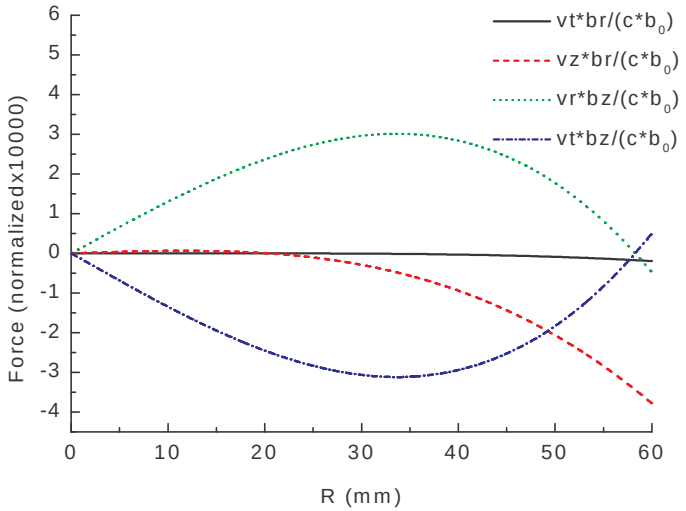
18.Radial magnetic self-field for a KV beam.



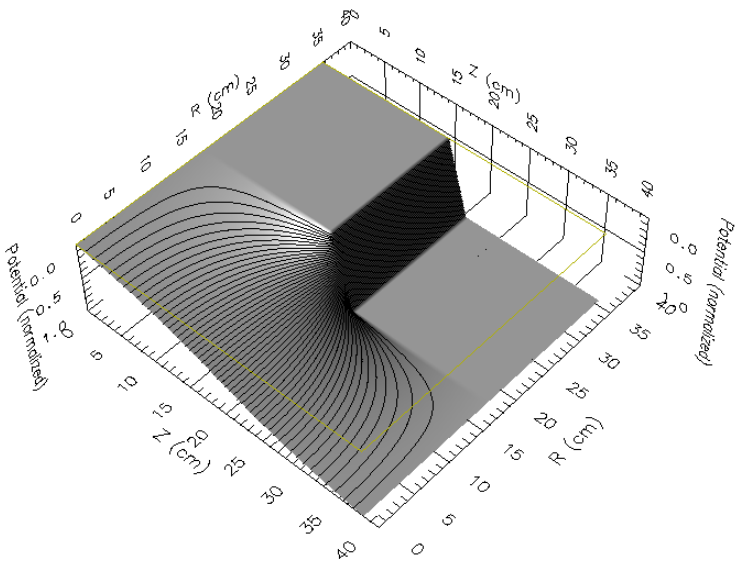
19.Azimuthal magnetic self-field for a KV beam.



20.Longitudinal magnetic self-field for a KV beam.



21.Some of the forces acting on the particles of a KV beam (in units normalized to the speed of light c and the external magnetic fields B_0).

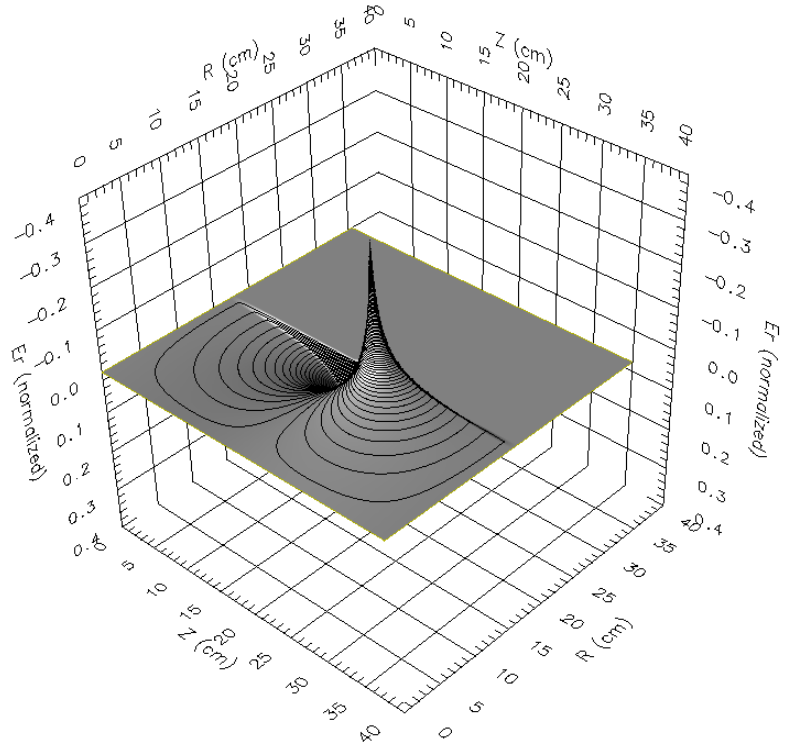


22.Potential field in the gap region. For clarity, the axis displaying the potential value has been reversed.

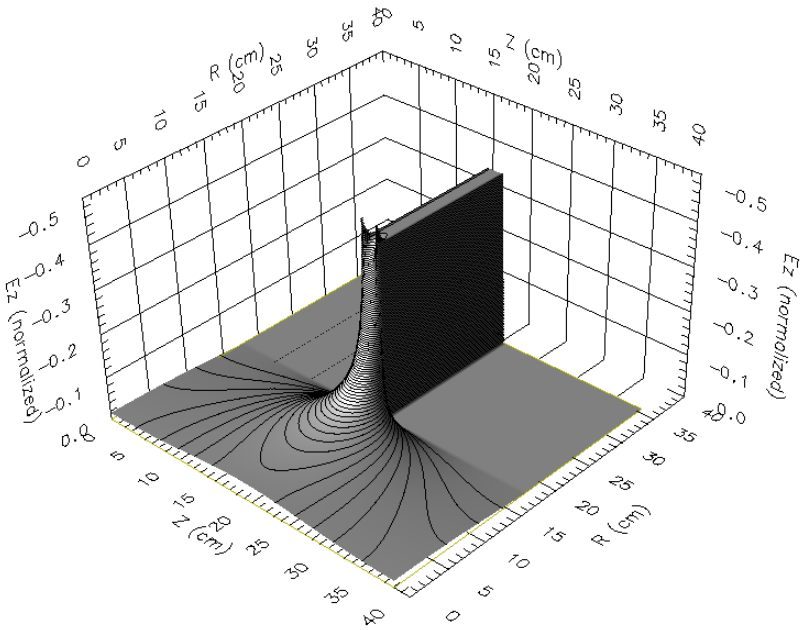
B Example of a typical gap field calculation

We have applied the multigrid algorithm to a typical gap for DARHT, where $\delta z_{gap} = 2.54$cm and R_{pipe}=17.78cm. We have chosen to have $2^6 + 1 = 65$ grid lines to describe R_{pipe}, $2^3 + 1 = 9$ grid lines to describe δz_{gap} and $n = 7$ so that the finest grid has $(2 \times 64 + 1) \times [(2 \times 7 + 1) \times 8 + 1] = 129 \times 121$ grid points and has an extension of $35.56cm \times 38.10cm$. The choice of $n = 7$ is not intended to fit a particular design of DARHTfiit may vary for an actual simulation with a DARHT lattice.

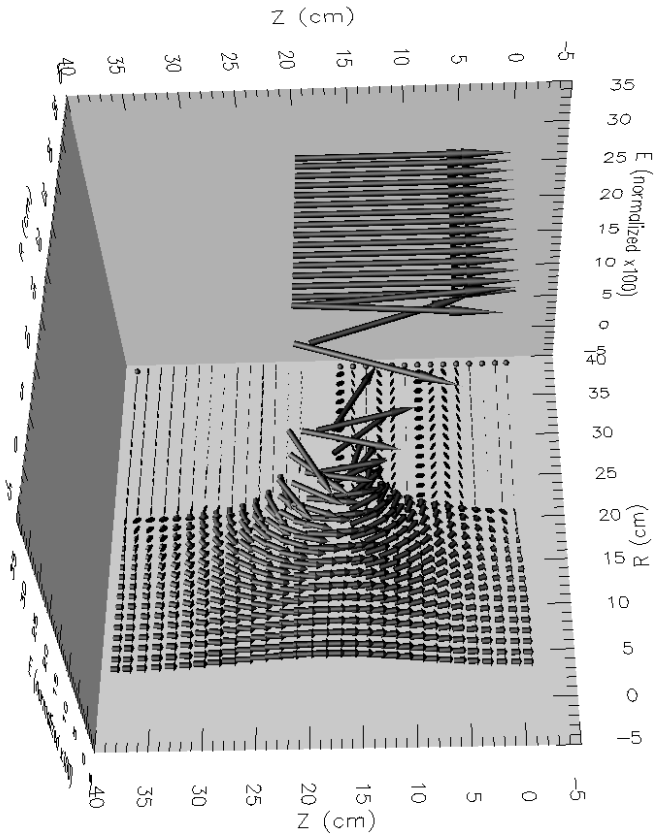
The potential and electric fields computed by the algorithm are plotted in Figures 22, 23, 24 and 25. We observe that the field extends far away from the gap location in the longitudinal direction, justifying our choice of an extended grid in this direction. We also observe in Figures 23 and 25 that the radial component of the field has a non-negligible value (relative to the longitudinal field magnitude) for positions which are off-axis near the gap position.



23.Radial electric field.



24.Longitudinal electric field.



25.Electric field. The directions and magnitude of the arrows are the ones of the fields. For clarity, a third axis as been created so that the origin of the arrow on this axis is proportional to the magnitude of the field (normalized x100).

C Input files sample

- general purpose init file

```
2.0 MV beam fiparameters from LSP July 99
tuned lattice for comparison with LSP
$
&BEAM
current = 2000.
energy_mev = 3.2
abeam = 0.081
rprime = 0.00445
emit_norm = 151e-6
npart = 4096
xoffset=0.00
yoffset=0.00
xpoffset = 0.e-3
ypoffset = 0.e-3
distribution = 'SG'
l_adjust_init_rot = .f.
l_init_pot_depress = .f.
l_ReadEgunOutput = .f.
l_ReadLSPOutput = .t.
egun_filename = '../inp/egun061699.dat'
LSP_filename = '../inp/lsp1.dat'
par_dump_file = 'NOT SET'
old_par_dump_file = 'NOT SET'
&END
&BEAM_TEST_NAMELIST
npart_test = 9
xtest = 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09
ytest = 0.000 0.00 0.0 0.0 0.0 0.00 0.00 0.00 0.000
vxtest = 0.
vytest = 0.
&END
&SIM_PARAM
lgraphic = .t.
l_opendx = .t.
nr = 500
l_paraxial = .f.
l_self_btheta = .t
l_self_bz = .t
l_self_br = .t
l_self_er = .t
l_self_ez = .t
l_self_envelope = .f.
l_image_fields = .f.
l_gap_er = .t.
gap_calc_method = 'multigrid'
```

```
zstart = 1.45
zmax = 50.
zstep = 0.005
dzphaseplot = 5.
npart_phaseplot = 4096
dzhist = 0.05
phaseplot_type = 'thermal,radial'
phaseplot_save_particles = .t.
l_save_hist = .t.
hist_dir = '../runs/run2'
save_hist_vars = 'all'
nzfidsave = 0
zfidsave = 1.450004
nzfidsnapshot = 6
zfidsnapshot = 1.45 6.45 11.45 16.45 21.45 36.45
&END
```

- lattice file

```
&lat_list
env_magfile = '../lat/jun99h2.mag'
l_cm = .true.
soldef_length = 40.64 10.16 40.64 19.91 10.00
soldef_r_inner = 13.97 13.97 19.05 18.69 44.76
soldef_r_outer = 16.51 16.51 21.59 19.91 47.30
soldef_n_layers = 1 1 1 1 1
soldef_name(1:5) = 'STANDARD','INTERCELL','INJECTOR CELL','ANODE COIL','VALVE
COIL'
gapdef_length = 2.54 2.54
gapdef_rmax = 17.78 12.7
gapdef_name(1:2) = 'INJ_GAP', 'STD_GAP'
gap_type = 8*1, 80*2
gap_z = 88*51.435
gap_z(1) = 210.28
gap_z(9)=450.615 gap_z(17)=94.615
gap_z(25)=94.615 gap_z(33)=94.615
gap_z(41)=94.615 gap_z(49)=94.615
gap_z(57)=94.615 gap_z(65)=94.615
gap_z(73)=94.615 gap_z(81)=94.615
gap_mev = 8*0.175, 80*0.19318
nwall = 2
rwall = 17.78 12.7
zwall = 0. 600.0
/END
```

- magnets file

```
jun99h2.mag 1999 07 07 13:26:51
&magnets
ns = 104
solnam(1:3) = "BUCK" "AND1" "AND2"
bsi(1:3) = -6486. 4494. 4494.
izs(1:3) = 0 0 0
zs(1:3) = -26.955 62.005 91.835
ltype(1:3) = 4 4 4
solnam(4) = "VAL1"
bsi(4) = 12700.
izs(4) = 0
```

zs(4) = 130.780
ltype(4) = 5
solnam(5:8) = ”J101” ”J102” ”J103” ”J104”
bsi(5:8) = 3750. 4440. 5130. 5820.
izs(5:8) = 0 1 1 1
zs(5:8) = 184.995 51.435 51.435 51.435
ltype(5:8) = 3 3 3 3
solnam(9:12) = ”J105” ”J106” ”J107” ”J108”
bsi(9:12) = 6510. 10800. 12000. 15000.
izs(9:12) = 1 1 1 1
zs(9:12) = 51.435 51.435 51.435 51.435
ltype(9:12) = 3 3 3 3
solnam(13:16) = ”HCU1” ”HCU2” ”J201” ”J202”
bsi(13:16) = 38000. 32000. 24700. 12278.
izs(13:16) = 0 0 0 1
zs(13:16) = 655.035 865.035 995.035 51.435
ltype(13:16) = 1 1 1 1
solnam(17:20) = ”J203” ”J204” ”J205” ”J206”
bsi(17:20) = 28571. 29857. 31143. 32429.
izs(17:20) = 1 1 1 1
zs(17:20) = 51.435 51.435 51.435 51.435
ltype(17:20) = 1 1 1 1
solnam(21:22) = ”J207” ”J208”
bsi(21:22) = 33714. 35000.
izs(21:22) = 1 1
zs(21:22) = 51.435 51.435
ltype(21:22) = 1 1
solnam(23) = ”I23”
bsi(23) = 27600.
izs(23) = 1
zs(23) = 47.308
ltype(23) = 2
solnam(24:27) = ”J301” ”J302” ”J303” ”J304”
bsi(24:27) = 36000. 36429. 36857. 37286.
izs(24:27) = 1 1 1 1
zs(24:27) = 47.308 51.435 51.435 51.435
ltype(24:27) = 1 1 1 1
solnam(28:31) = ”J305” ”J306” ”J307” ”J308”
bsi(28:31) = 37714. 38143. 38571. 39000.
izs(28:31) = 1 1 1 1
zs(28:31) = 51.435 51.435 51.435 51.435
ltype(28:31) = 1 1 1 1
solnam(32) = ”I34”
bsi(32) = 30492.
izs(32) = 1
zs(32) = 47.308
ltype(32) = 2
solnam(33:36) = ”J401” ”J402” ”J403” ”J404”
bsi(33:36) = 39500. 39857. 40214. 40571.
izs(33:36) = 1 1 1 1
zs(33:36) = 47.308 51.435 51.435 51.435
ltype(33:36) = 1 1 1 1
solnam(37:40) = ”J405” ”J406” ”J407” ”J408”
bsi(37:40) = 40929. 41286. 41643. 42000.
izs(37:40) = 1 1 1 1

```
zs(37:40) = 51.435 51.435 51.435 51.435
ltype(37:40) = 1 1 1 1
solnam(41) = ”I45”
bsi(41) = 33360.
izs(41) = 1
zs(41) = 47.308
ltype(41) = 2
solnam(42:45) = ”J501” ”J502” ”J503” ”J504”
bsi(42:45) = 42500. 42857. 43214. 43571.
izs(42:45) = 1 1 1 1
zs(42:45) = 47.308 51.435 51.435 51.435
ltype(42:45) = 1 1 1 1
solnam(46:49) = ”J505” ”J506” ”J507” ”J508”
bsi(46:49) = 43929. 44286. 44643. 45000.
izs(46:49) = 1 1 1 1
zs(46:49) = 51.435 51.435 51.435 51.435
ltype(46:49) = 1 1 1 1
solnam(50) = ”I56”
bsi(50) = 35700.
izs(50) = 1
zs(50) = 47.308
ltype(50) = 2
solnam(51:54) = ”J601” ”J602” ”J603” ”J604”
bsi(51:54) = 45500. 45857. 46214. 46571.
izs(51:54) = 1 1 1 1
zs(51:54) = 47.308 51.435 51.435 51.435
ltype(51:54) = 1 1 1 1
solnam(55:58) = ”J605” ”J606” ”J607” ”J608”
bsi(55:58) = 46929. 47286. 47643. 48000.
izs(55:58) = 1 1 1 1
zs(55:58) = 51.435 51.435 51.435 51.435
ltype(55:58) = 1 1 1 1
solnam(59) = ”I67”
bsi(59) = 37620.
izs(59) = 1
zs(59) = 47.308
ltype(59) = 2
solnam(60:63) = ”J701” ”J702” ”J703” ”J704”
bsi(60:63) = 48480. 48714. 48929. 49143.
izs(60:63) = 1 1 1 1
zs(60:63) = 47.308 51.435 51.435 51.435
ltype(60:63) = 1 1 1 1
solnam(64:67) = ”J705” ”J706” ”J707” ”J708”
bsi(64:67) = 49357. 49571. 49786. 50000.
izs(64:67) = 1 1 1 1
zs(64:67) = 51.435 51.435 51.435 51.435
ltype(64:67) = 1 1 1 1
solnam(68) = ”I78”
bsi(68) = 39804.
izs(68) = 1
zs(68) = 47.308
ltype(68) = 2
solnam(69:72) = ”J801” ”J802” ”J803” ”J804”
bsi(69:72) = 50500. 50714. 50929. 51143.
izs(69:72) = 1 1 1 1
```

zs(69:72) = 47.308 51.435 51.435 51.435
ltype(69:72) = 1 1 1 1
solnam(73:76) = ”J805” ”J806” ”J807” ”J808”
bsi(73:76) = 51357. 51571. 51786. 52000.
izs(73:76) = 1 1 1 1
zs(73:76) = 51.435 51.435 51.435 51.435
ltype(73:76) = 1 1 1 1
solnam(77) = ”I89”
bsi(77) = 41409.
izs(77) = 1
zs(77) = 47.308
ltype(77) = 2
solnam(78:81) = ”J901” ”J902” ”J903” ”J904”
bsi(78:81) = 52500. 52714. 52929. 53143.
izs(78:81) = 1 1 1 1
zs(78:81) = 47.308 51.435 51.435 51.435
ltype(78:81) = 1 1 1 1
solnam(82:85) = ”J905” ”J906” ”J907” ”J908”
bsi(82:85) = 53357. 53571. 53786. 54000.
izs(82:85) = 1 1 1 1
zs(82:85) = 51.435 51.435 51.435 51.435
ltype(82:85) = 1 1 1 1
solnam(86) = ”I910”
bsi(86) = 42900.
izs(86) = 1
zs(86) = 47.308
ltype(86) = 2
solnam(87:90) = ”J1001” ”J1002” ”J1003” ”J1004”
bsi(87:90) = 54250. 54429. 54607. 54786.
izs(87:90) = 1 1 1 1
zs(87:90) = 47.308 51.435 51.435 51.435
ltype(87:90) = 1 1 1 1
solnam(91:94) = ”J1005” ”J1006” ”J1007” ”J1008”
bsi(91:94) = 54964. 55143. 55321. 55500.
izs(91:94) = 1 1 1 1
zs(91:94) = 51.435 51.435 51.435 51.435
ltype(91:94) = 1 1 1 1
solnam(95) = ”I1011”
bsi(95) = 44240.
izs(95) = 1
zs(95) = 47.308
ltype(95) = 2
solnam(96:99) = ”J1101” ”J1102” ”J1103” ”J1104”
bsi(96:99) = 55750. 55929. 56107. 56286.
izs(96:99) = 1 1 1 1
zs(96:99) = 47.308 51.435 51.435 51.435
ltype(96:99) = 1 1 1 1
solnam(100:103) = ”J1105” ”J1106” ”J1107” ”J1108”
bsi(100:103) = 56464. 56643. 56821. 57000.
izs(100:103) = 1 1 1 1
zs(100:103) = 51.435 51.435 51.435 51.435
ltype(100:103) = 1 1 1 1
solnam(104) = ”I11FF”
bsi(104) = 49800.
izs(104) = 1

```
zs(104) = 48.000
ltype(104) = 2
/END
```

D A typical run

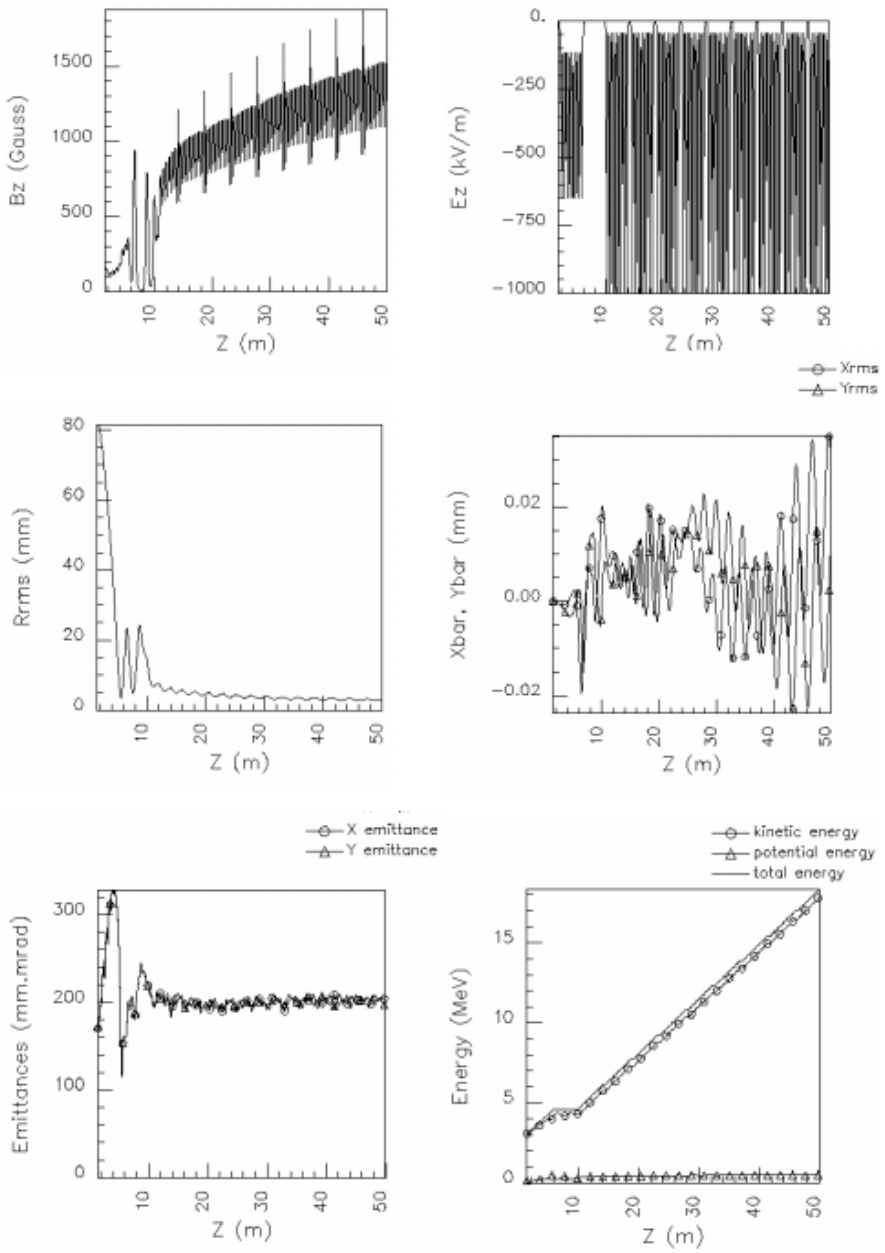
A run was performed with the input files presented in the preceding appendix. On a Pentium pc (400MhZ), the run was performed in about 6min and 40sec. The results of the run are given on Fig.26.

[1] J.-L. Vay, ”Multigrid field solver for calculation of gap fields in the AMBER PIC code”, LBNL report, Oct. 1999

[2] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, ”Numerical Recipes in FORTRAN”, Second Edition, Cambridge University Press, 1992, pp. 862-880

[3] W. Hackbush, ”Multigrid Methods and Applications”, New-York: Springer-Verlag

[4] C. K. Birdsall and A. B. Langdon, ”Plasma Physics via Computer Simulation”, New-York: McGraw-Hill, 1991



26.Results of the example run (external magnetic field on axis, external electrostatic field on axis, rms radius history, beam centroid history, beam emittances history, beam energies history).